CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Understanding and Modelling Behavioural Requirements: an Exploratory Study

Bachelor of Science Thesis in Software Engineering and Management

Marco Trifance
Ivo Vryashkov

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Understanding and Modelling Behavioural Requirements: an Exploratory Study

Marco Trifance
Ivo Vryashkov

Supervisor: Grischa Liebel
Examiner: Rodi Jolak

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

Understanding and Modelling Behavioral Requirements: an Exploratory Study

Marco Trifance and Ivo Vryashkov
Department of Computer Science and Engineering
University of Gothenburg and Chalmers University of Technology
Gothenburg, Sweden
marco.trifance@gmail.com, ivo.vryashkov@gmail.com

***Abstract*—Clear understanding of system requirements is necessary to achieve quality in the architectural design and in the development process of a software system. Several studies focus on the comprehensibility of graphical modelling languages. Contributions to other areas in Software Engineering use empirical investigation to explore how individuals approach collaborative learning tasks in different phases of software development. This paper describes an exploratory case study we conducted with 10 undergraduate students to investigate how subjects approach modelling of system requirements. We used the method of constructive interaction to identify the most common difficulties and to explore whether different requirements specification formats affect the approach of the subjects. We observed that the most common difficulties were related to misuse of UML syntax elements. Furthermore, our findings suggest that the approach of the subjects is affected by the completeness of the requirements specification they use.**

***Keywords*-requirements understanding, requirements modelling, constructive interaction**

I. Introduction

Requirements Engineering (RE) is a fundamental phase in Software Engineering (SE) [1]. SE is a complex activity often involving a multiplicity of actors. Complete and communicative specifications of requirements are necessary to produce a good architectural design, which in turn increases the quality of the development process and of the system under development [2].

Academic literature has drawn particular attention to the area of comprehensibility of different requirements specification techniques. With the purpose of enhancing the understanding and communication of requirements among different stakeholders, a branch of research has focused on the creation and improvement of requirements modelling languages [3], [4]. A number of studies use controlled experiments to compare the comprehensibility of different modelling languages [5], [6], [7]. In other areas of SE, e.g. program comprehension [8], [9], recent studies have investigated how practitioners approach learning tasks in different phases of software development. However, there is to our knowledge no related work exploring how individuals translate requirements into a model and how this affects their learning about the requirements during the process.

The purpose of this study is to explore the process of understanding and modelling requirements. Our goal is to investigate how students proceed when creating requirements models and which difficulties they experience. We also explore the differences in the modelling approach when using different levels of details in the requirements specifications. To do so, we address the following research questions (RQs):

R.Q. 1: How do student subjects approach requirements modelling?

R.Q. 2: Which difficulties do student subjects experience when creating requirements models?

R.Q. 3: What are the differences in the modelling process depending on the input format and completeness of the requirements?

To address our research questions we use analysis of qualitative data collected from a controlled environment. Our data collection and analysis procedures are based on the method of *constructive interaction* [10], which was introduced in the area of cognitive science to overcome some of the limitations that existing literature has associated with think-aloud protocols [11].

This study is designed as a replication study of an ongoing research project conducted by Liebel [12], [13]. At the time we conducted this study, the parent project was still undergoing data collection activities and data analysis was not yet completely defined. In this perspec-

tive, we provide three contributions. First, we provide a description of the behaviour observed in subjects and a list of the most common difficulties encountered while performing modelling tasks. Our second contribution consists of a list of codes to support the analysis of textual data in the parent project. Our final contribution is represented by the instruments–i.e. requirements specifications–we will create in our replication [12].

The remainder of this paper is structured as follows: in Section II we present a literature review of the related work to our study. We outline our research methodology in Section III. In Section IV we present the results from our study which are later discussed in Section V. In Section VI we summarize the main threats to the validity of this study. Finally, in Section VII, we present our conclusions.

II. Related Work

Over the past two decades, a number of studies have contributed to the area of requirements modelling from different perspectives. Some authors have suggested new variations of requirements modelling languages with the purpose of enhancing communication among stakeholders with different areas of expertise, e.g. [3] and [4]. Helming et al. propose the Unified Requirements Modelling Language (URML) to provide a homogeneous and comprehensible representation of requirements aimed to support interdisciplinary collaboration [3]. Jureta et al. propose *Techne*, an abstract requirements modelling language (RML) to serve as a basis on which other RMLs can be built [4].

A number of experimental studies compare different requirements specification languages with respect to comprehensibility. Otero and Dolado conduct an experiment over 31 undergraduate students to compare the comprehensibility of UML state machine, sequence and collaboration diagrams in real-time and management information systems [7]. Their results suggest that sequence diagrams display better comprehensibility for real-time systems than for management information systems. Abrahão et al. run a series of experiments to show that enhancing textual requirements specification with UML sequence diagrams increases comprehensibility [6]. Liebel and Tichy use a controlled experiment over 22 undergraduate students to compare Modal Sequence Diagrams (MSD) and Timed Automata (TA) [5]. The results show no significant difference with respect to comprehensibility. However, the authors report how the subjects provided with MSD answered significantly more questions.

Recent studies in areas related to SE explore how individuals approach learning tasks in activities related to software design [14] and development [8], [9]. Stikkolorum et al. use an online experiment with 120 student-pairs to identify common problems encountered and the main strategies adopted by students during class diagrams design tasks [14]. The authors found that the problems the students have are related to wrong use of UML elements and adding unnecessary elements to class diagrams. Sillito et al. conduct two studies to identify the most common questions asked by developers during programming change tasks [9]. Their contribution includes a catalog of 44 types of questions classified in four top-level categories, namely finding focus points, expanding focus points, understanding a subgraph and questions over groups of subgraphs. Duala-Ekoko and Robillard use a controlled environment to investigate how developers seek information when using unfamiliar APIs [8]. They observe 20 graduate students approaching programming tasks involving external APIs to identify the most common questions and difficulties encountered by the subjects.

To our knowledge, there is no related work investigating how practitioners understand and model behavioural requirements. With a similar intent to Duala-Ekoko and Robillard, in this study we use a controlled environment to investigate how students approach requirements modelling and to identify the main difficulties they encounter.

III. Methodology

A. Research Method

To address our research questions we conducted an exploratory case study [15]. We follow a qualitative approach based on the method of constructive interaction [10]. Constructive interaction consists of observing and analyzing the interactions between two individuals while they cooperate in collaborative problem solving activities. The basic idea is that the identification of constructive interactions allows to gain information on the reasoning process of the subjects [10].

Because of its convenience in terms of time and cost of data collection procedures, we used a controlled environment to observe the modelling activities of the students. Although the use of a controlled environment might appear in contrast with qualitative data analysis driven by exploratory purposes, academic literature has provided an increasing number of studies following this type of approach [8], [16], [17].

B. Participants

We invited 10 students who performed the task of modelling behavioural requirements in groups of two. The students were 3rd year Software Engineering undergraduates enrolled in the Software Engineering and Management program at the University of Gothenburg.

We used convenience sampling for selecting the participants in our study [18]. More specifically, all participants were selected from the same university we were enrolled at the time we conducted this study. This allowed us to save time and effort in conducting the necessary data collection activities.

C. Study Setup

We conducted the study in multiple sessions, each one involving a group of two student subjects. In each session the researchers observe the pair of students cooperate to create UML (Unified Modeling Language [19]) state machines from a set of behavioural requirements specified in a textual format.

UML state machines are utilized for modelling the dynamic aspects of systems. In particular, they are best suited for specifying the behaviour of real-time systems [20]. UML state diagrams focus on the event-governed behaviour of an object where flow of control transitions from one state to another. In other words, state machines describe the possible sequences of states that an object can go through during its lifecycle in reaction to certain events and what actions are taken when events occur [21].

The choice of UML state machines was mainly driven by two factors. First, UML is widely accepted as the standard in SE [22]. Second, we knew that the participants in our study were familiar with UML state machines (i.e. the students have undertaken courses covering the use and notation of UML state machines as part of their university program).

Each subject was given a cheat sheet on UML state machines one week before the scheduled study session (available in Appendix D). The purpose of the cheat sheet was for the students to refresh their knowledge and memories about drawing state diagrams using UML syntax. However, during the actual modelling iterations, the participants were not allowed to use the cheat sheet as reference. In addition, any other external help such as laptops, mobile phones, was not allowed during the study sessions.

We created both User Requirements Specifications (URS) [23] and more detailed System Requirements Specifications (SRS) [23] for two different systems: an elevator and a drying machine (see Appendix A). This gave us a total of four requirements specification documents. We produced two different formats for each system with the intent of creating requirements specifications that would present a different level of completeness. We used these different requirements formats to investigate how students approach requirements modelling in two different scenarios. In the first scenario (from now on called *iterative approach*) the students are provided with a vague description of the system requirements (the URS), while in the second scenario (from now on called *specific approach*) the students are provided with a more detailed description of the intended behaviour of the system (the SRS).

At the beginning of each session the subjects are given a short introduction where they are explained the rules defining the study. In order to avoid researcher bias, the introduction is given as a video[1]. The remaining time is articulated in two phases, each one dedicated to the modelling of a different set of requirements.

We organized the two modelling phases to follow both the iterative and the systematic approach. For example, if the subjects are provided with URS (iterative approach) for ”system A” in phase one, they receive SRS (specific approach) for ”system B” in phase two. To be able to compare the two approaches, the combination and order of system and requirements specification format (URS or SRS) is inverted each session [24].

During the modelling phases, the researcher acts as the customer, meaning that the students can use the time between the modelling iterations to ask for clarifications about the requirements specifications. However, interactions between the subjects and the customer (researcher) are not allowed during the modelling iterations. When answering the questions, the researcher provides information related to the system (i.e. problem domain), without commenting on the quality or correctness of the model (the solution) produced by the students. Also, the researcher does not answer any question related to UML syntax or modelling conventions. Concerning the syntax, students are encouraged to use elements and notation according to the rules defined within UML. However, since measuring the UML knowledge of the students is not the primary focus of this study, modelling at a sketching level is also accepted. More specifically, subjects are allowed to declare their own notation to handle those cases where they do not remember the correct use of a given state machine element.

[1] https://youtu.be/Kv87fKsH7p0

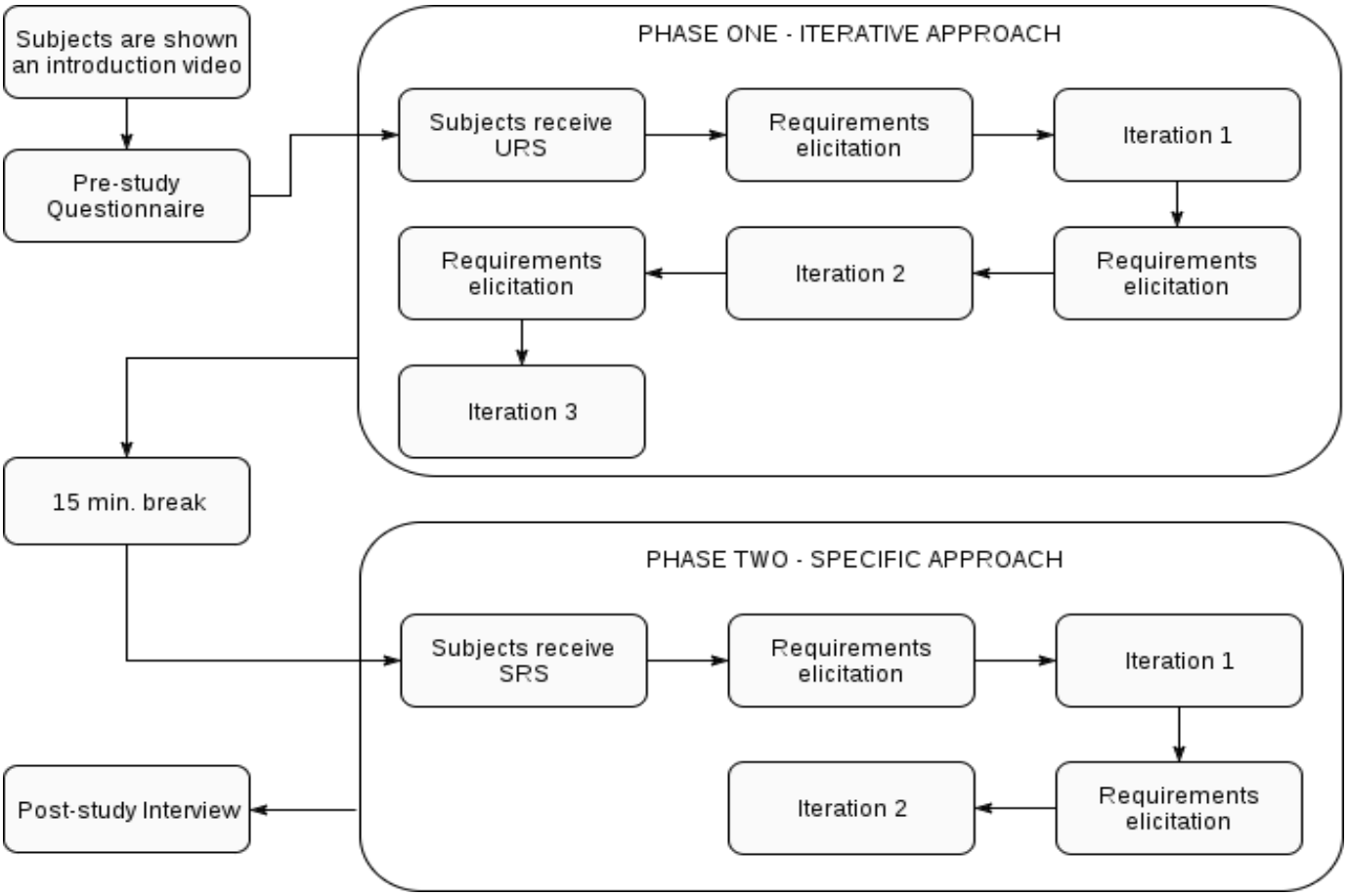


Fig. 1. Study session overview

The iterative approach consists of three iterations of 15 minutes each, while the specific approach has two iterations of 20 minutes. The intention behind the different number and length of the iterations is to make the modelling phases more realistic. On the one hand, in the iterative approach, the subjects receive a set of requirements (URS) that is lacking in details and completeness. Hence, they have more iterations with a reduced time frame which allows more frequent interactions with the customer (researcher). On the other hand, in the specific approach, the participants receive detailed and complete specifications of the system. Because of this level of detail of the requirements, the modelling iterations are designed to be longer and less frequent. This gives the subjects fewer opportunities to talk to the customer but more time to model the system. We are aware that the above mentioned differences lower the control of the environment. However, we believe that the exploratory nature of the study makes reduced control acceptable to our purposes.

Figure 1 presents an overview of the session process in our study. The scenario depicts the case where the subjects begin in phase one with modelling the URS following the iterative approach and then continue in phase two with modelling the SRS following the specific approach.

D. Data Collection

We used multiple data collection procedures to address our research questions. Before each session the participants fill in a pre-study questionnaire we use mainly for demographic purposes [25]. We use the collected data to show the general experience of the participants in software engineering, requirements understanding and requirements modelling. A copy of the questionnaire is provided in Appendix B.

The main source of data consists of audio and video recordings from the study sessions conducted with the students. In addition to the recorded material, the researchers also took notes of the process and interactions during the modelling iterations [26].

After each session we conducted a semi-structured

Code	Title	Explanation/Usage
p	**Propose**	1. Propose a concrete solution or procedure. - ex. *"Let's start with idle"* or *"Let's read first, then we draw"*.
		2. Suggest how the system or the model is working (indicating modal verbs: can, will, is able to).
		3. Subject draws part of the model. - ex. *"It goes from state X to state Y"* while drawing corresponding transition in the model.
q	**Question**	1. Ask a question or request additional information. - ex. *"What do you mean here?"*
		2. **NOT** when subject is questioning the validity of something (use "**c**" instead).
c	**Criticise**	1. Question the current solution or the validity of something. - ex. *"Do you think we should really do like this?"* or *"I think this is not correct"*
		2. Includes realization that a previous statement is wrong - ex. *"Ooohhh, this is wrong"*
		3. Use conservatively. Use "**q**" if interaction is not of type 1 or 2.
m	**Mediate**	1. Recommend to ask something to the researcher/stakeholder. - ex. *"Maybe we should ask this in the break"* or *"I think this is something we need to ask"*
a	**Acknowledge**	1. Confirm or acknowledge something. - ex. *"Yeah."* or *"Yes, I think this is good."*
r	**Reason**	1. Reason about how the system or the model should behave. - ex. *"And then when car comes it should go to state X"*. (Indicating modal verbs: should, could, might, may)
		2. Use according to subject activity. If related to the model, use when the solution is already existing (Use "**p**" if the subject is drawing the solution). - ex. *"It goes from state X to state Y"* while pointing at transition in the model.
n	**Lack of knowledge**	1. Subject declares lack of knowledge or information on a specific topic. - ex. *"I don't know how this is supposed to work"*
		2. **Use conservatively.** Statements like *"I don't know"* can be common in informal talk. Make sure the intention of the subject can be reconducted to case 1.
n/a	**No code**	1. Empty rows containing meta information. - ex. comment (Both subjects draw at the same time)
		2. Incomplete, unclear sentences. - ex. *"mmm... we should..."*

TABLE I
PROCESS CODES

interview with the students [27]. The purpose of the interviews is to give us more insights on the approach followed by the subjects and to support the findings from the modelling sessions. The questions included in the interview are provided in Appendix C.

E. Data Analysis

Data analysis was carried out iteratively and in parallel with data collection activities. This allowed us to follow an editing approach [27], i.e. we started with a small set of themes which was updated and expanded according to new information as data collection proceeded.

We used data from the pre-study questionnaire to identify differences in the background and level of experience of the participants. We addressed our RQs mainly through the analysis of the recordings from the study sessions and of the post-study interviews.

Here we briefly summarize the steps we followed for the analysis of the recordings from each study session. The remainder of this section provides a detailed description of each of these steps. We created verbatim transcriptions of the audio recordings from the modelling sessions. We made use of the video recordings to enrich the transcriptions with relevant information on the behaviour of the subjects, e.g. "subject A starts drawing state X" [26]. The resulting transcriptions were then coded separately by both researchers.

With the intent of mitigating researcher bias and improve the reliability of our data, we measured the level of agreement between the codes assigned by the two researchers. This means that the transcription of each single modelling iteration has been coded iteratively until the desired level of agreement was observed. Once the data was validated, we randomly selected one of the two coded transcriptions, we marked it as reliable data and added it to the dataset for our final data analysis.

In the final step of our analysis we triangulated the information extracted from coded transcriptions with data from the post-study interviews and used inductive category building [28] to answer our RQs.

Transcriptions

In Appendix E we provide an excerpt of one of our

Code	Title	Explanation/Usage
m	**Model**	1. The interaction includes terms that identify elements that are present in the solution modeled by the subjects. - ex. *"here this guard evaluates to true"*, *"this transition needs a trigger"*
		2. Subject is drawing or pointing at the model.
		3. **NOT** when subject is sketching the system on the whiteboard. Use "**s**" instead.
s	**System**	1. Subject is discussing the system behavior without mentioning model elements.
		2. Subject is reading the requirements specification document.
		3. USE conservatively. Use "**m**" unless it is clear that it is 1 or 2.
u	**UML Syntax**	1. Subject is discussing UML state machine modelling elements without referring directly to elements that are present in their specific solution. - ex. *"usually in state machines these cases are handled with nested states and choice nodes"*
		2. Interaction focuses on UML modelling rules and conventions, without referring to elements of the solution modeled by the subjects - ex. *"choice nodes are used to handle multiple conditions"*
		3. USE conservatively. Use "**m**" unless it is clear it is 1 or 2.
e	**Environment Settings**	1. The interaction is related to the study environment, rules, settings and tools. - ex. *"How long do we have left before the end of the iteration?"* or *"Can I have another pen?"*
p	**Procedure**	1. Subject discusses the procedure, tasks and activities to complete to produce the model. - ex. *"Ok, you draw, I read the requirements."*
		2. **NOT** when the interaction refers to elements of to the model directly. Use "**m**" instead.
n/a	**Other**	1. None of the above.
		2. Metadata.

TABLE II
TOPIC CODES

coded transcriptions. We used a spreadsheet to divide the protocol in a sequence of interactions (column three) between the two observed subjects (column two). Single interactions were then split whenever the transcriber noticed an interruption in the sentence, a change in the topic (e.g. model, system requirements, environment) or in the intentions of the subject (e.g. proposing, criticizing). The first column was used to mark the minute when the interaction was recorded. Column number six displays the notes and comments that were added by the transcriber with the use of the video recordings. The transcriber notes are mainly intended to describe the physical motions and actions of the subjects (e.g. drawing, reading, mimicking), and more generally to provide all the information that could not be captured in the audio recordings [26].

Coding

During the coding of the observed interactions we focused on two distinct dimensions: *process* and *topic*. With process coding we attempt to capture the way the subjects proceed, how they reason and cooperate to model their solutions. With topic coding we intend to identify the context discussed by the participants, e.g. system requirements, study environment, model.

The idea behind process coding is to compare the sample distributions of the process codes over different groups to investigate whether they tended to display common approaches and patterns to modelling of behavioural requirements (RQ1). In a similar perspective, the comparison of the distributions of the process codes over the two alternative requirements specification formats–iterative (URS) and specific (SRS) approaches–allows us to explore how different levels of details in the specification affects the approach of the students (RQ3).

In our analysis over the process dimension we made use of a set of codes defined in the replicated study [12]. These codes were inspired by the works by Soller et.al [29], Miyake [10] and Baker [30]. Based on these observations during data collection, we decided to extend this list by introducing a new category of process interaction ("n" – lack of knowledge). Table I provides the definitions of the codes we used to classify the interactions over the process dimension.

Some of the process codes were relatively easy to detect, and therefore less prone to create disagreement in the interpretation by different researchers. For example, it is safe to say that this was the case for interactions of type "m" (mediate) or "n" (lack of knowledge). However, some boundaries were less clear than others, especially when the tone of the conversation tended to be more informal. A valid example is the distinction between

	Iterative Approach			Specific Approach	
Coding Iteration	Iteration 1	Iteration 2	Iteration 3	Iteration 1	Iteration 2
1st	0.575	0.657	0.629	0.372	0.654
2nd	**0.683**	**0.725**	**0.834**	0.645	0.653
3rd	-	-	-	**0.676**	**0.674**

TABLE III
CODING ITERATIONS

codes ”r” (reason) and ”p” (propose). To facilitate this specific distinction, we created some rules of thumb based on the physical actions of the participants which were described in the researcher comments. For this specific case, actions like drawing a state, declaring a variable, writing a guard, and all concrete and active contributions to the model were classified as ”p”. Conversely, all interactions where the subject was pointing at the model discussing elements already present in the solution were classified as ”r”.

Topic coding was introduced to identify the context discussed in each interaction. The idea behind topic coding is that its use in conjunction with process coding would provide us with relevant details on the behaviour of the subjects. For example, by knowing which topic is being addressed in a specific interaction we could extract information on the nature of the difficulties encountered by the subjects (RQ2). Topic coding also allowed us to measure the extent to which subjects discussed system requirements (problem domain) or model design (solution domain). This information was later used to reflect on how the approach of the subjects changed in relation to the level of details in the requirements specifications (RQ3).

Based on our observations, we identified five main areas discussed by the subjects during the modelling iterations. Table II provides the definitions of the codes we used to classify the interactions over the topic dimension. As in the case for process codes, some of the topic categories were more easily identifiable than others. The most difficult distinction in this case was in determining whether the subjects were discussing the model or the system. For this specific case we made intensive use of the transcriber notes to consider the physical actions of the subjects in our interpretation of the interactions. For example, all interactions associated with concrete modifications of the model (e.g. subjects add or remove a state) were coded as ”m”.

Measuring the Level of Agreement

In order to measure the intercoder agreement on our coded data we used the Krippendorff’s alpha [31]. The Krippendorff’s alpha is an agreement coefficient that can handle categorical data and small sample sizes, which made it ideal for our case (two modelling iterations were particularly shorter than the others, counting 33 and 30 interactions respectively). We set our minimum threshold to an alpha on 0.667. This specific value was suggested by Krippendorff as sufficient to infer reliability of data within research studies where tentative conclusions are acceptable [31].

As we mentioned above, we proceeded by coding the transcriptions from each single modelling iteration in an iterative manner until we observed a value of the alpha above 0.667. Table III displays the coding iterations and the obtained alpha levels for one of the early sessions in our study (the cells marked with a hyphen denote iterations not used as the alpha level was already achieved). In cases, such as the specific approach, the coding of the transcriptions were iterated three times until the desired level of agreement was reached. This phase of the data analysis was particularly helpful to the identification of new codes to be included in our analysis and in reshaping the definitions and boundaries of those already in use. As a result, the coding process of the later sessions in our study became more efficient and coherent, and in certain cases the desired alpha level was achieved from the first coding iteration.

Data Triangulation

In the final step of our analysis we triangulated the information we extracted from our observations with the answers the students provided in the post-study interviews. Based on our observations, we tried to identify common patterns in the approach followed by the participants (RQ1). We also make use of the coded transcriptions to support our answers to RQ2 and RQ3. The joint analysis of both process and topic coding is also intended to produce insights on the difficulties encountered by the subjects (RQ2). We compare the frequencies of the process and topic codes over different approaches (iterative and specific approach) to investigate on how

Group	Participant	Industrial Experience	UML Knowledge[2]	FSM Knowledge[3]	Modelling Courses	Embedded Systems	Vertical Trans. Systems	Modelling Activities
G1	P1	0 years	intermediate	intermediate	TAD[4], SA[5], MDD[6]	basic	intermediate	weekly
	P2	0 years	intermediate	intermediate	TAD, SA	basic	intermediate	weekly
G2	P3	0 years	basic	basic	TAD, SA, MDD	intermediate	no knowledge	yearly
	P4	0 years	basic	basic	TAD, SA	basic	basic	yearly
G3	P5	0 years	basic	basic	TAD, SA	basic	no knowledge	yearly
	P6	6 months	intermediate	intermediate	TAD, MDD, SA	basic	basic	yearly
G4	P7	0 years	basic	basic	TAD, DP[7], SA, SP[8]	basic	no knowledge	yearly
	P8	0 years	basic	basic	TAD, SA, SP, TAV[9], PPPM[10]	basic	basic	yearly
G5	P9	0 years	intermediate	basic	TAD, MDD, TAV	basic	basic	yearly
	P10	0 years	basic	basic	TAV, SA	basic	basic	monthly

TABLE IV
PRE-STUDY QUESTIONNAIRE RESULTS

different requirements specification formats affect the modelling approach of the subjects (RQ3).

IV. RESULTS

This section includes the results we obtained during this study. We first present the information we collected from our pre-study questionnaire. In the second part we present the data from our coded transcriptions and we run statistical analysis on it to support our discussion in Section V.

Pre-study Questionnaire

We conducted five study sessions, involving a total of 10 students. Table IV summarizes the information we collected from the pre-study questionnaires. All participants were third year bachelor students within the program Software Engineering and Management at Gothenburg University. Two subjects were female. Almost all the subjects had no industrial experience in either software modelling, software development or requirements engineering. The only exception was one subject with an experience of 6 months. The participants have taken similar modelling courses, most of which are given at the university program in which the students are enrolled. Columns 4 and 5 show that subjects rated their prior knowledge in state machines as similar to their general knowledge of UML. One exception is a student who has taken a stand alone course in Design Patterns. Additionally, the knowledge of the subjects in the domains related to our study (embedded systems and vertical transportation systems) was diversified, ranging from no knowledge to intermediate knowledge. However, we stress that the information related to subject "knowledge" was given as personal rating by the subjects and therefore any difference must be interpreted with caution.

Study Sessions

During our sessions we collected a total of 331 minutes of recorded material, 265 of which from the modelling iterations and 66 from the post study interviews. Table V shows the length in minutes of each modelling iteration, divided per group and approach (specific and iterative). Cells marked with a dash symbol denote iterations that were not used by the subjects as they had already handed in their solution. Table VI shows the requirements specification documents that were handed in to the groups over the two phases.

[2]Possible choices – no knowledge; basic; intermediate; expert
[3]FSM – Finite State Machines
[4]TAD – Technical Analysis and Design
[5]SA – Software Architecture
[6]MDD – Model Driven Development
[7]DP – Design Patterns
[8]SP – Software Processes
[9]TAV – Test and Verification
[10]PPPM – Product, Projects and People Management

Approach	Specific			Iterative				
Iteration	**1st**	**2nd**	**Total Specific**	**1st**	**2nd**	**3rd**	**Total Iterative**	**Total**
Group 1	20	18	**38**	15	12	-	**27**	**65**
Group 2	20	12	**32**	15	11	-	**26**	**58**
Group 3	20	15	**35**	12	8	-	**20**	**55**
Group 4	16	5	**21**	15	-	-	**15**	**36**
Group 5	19	6	**25**	15	11	-	**26**	**51**

TABLE V
LENGTH OF THE MODELLING ITERATIONS IN MINUTES

Group	Phase One	Phase Two
G1	URS, Dryer	SRS, Elevator
G2	URS, Elevator	SRS, Dryer
G3	SRS, Dryer	URS, Elevator
G4	SRS, Elevator	URS, Dryer
G5	SRS, Dryer	URS, Elevator

TABLE VI
REQUIREMENTS SPECIFICATIONS

The values displayed in Table V show how all groups managed to submit their solutions within the time limitations we set for the modelling sessions. This is reinforced by the answers provided by the subjects to the post-study interviews, where all groups stated that they felt they had enough time to create their models. However, we observed clear differences in the total effort invested by different groups, ranging from a minimum of 36 to a maximum of 65 minutes. This different effort resulted in solutions displaying different levels of completeness and functionality.

We used the audio and video recordings from the study sessions to create written transcriptions of the interactions between the subjects. This step was carried out individually by one researcher. The transcriptions were later coded by both researchers individually, and then compared to measure the level of agreement between the two interpretations. This process resulted in a total of 3892 coded interactions, which were later reduced to 3639 after the removal of 253 interactions that had been classified as either non-constructive, incomplete or unclear, i.e. code "n/a" in both process and topic. Figures 2 and 3 use bar charts to display the frequencies of the observed interactions over the process and topic dimensions respectively.

Figures 2 and 3 are given with the purpose of providing a general overview of the interactions we observed during our study. By looking at both figures we see how students spent considerable effort discussing–mainly reasoning ("r"), acknowledging ("a") and proposing ("p")–

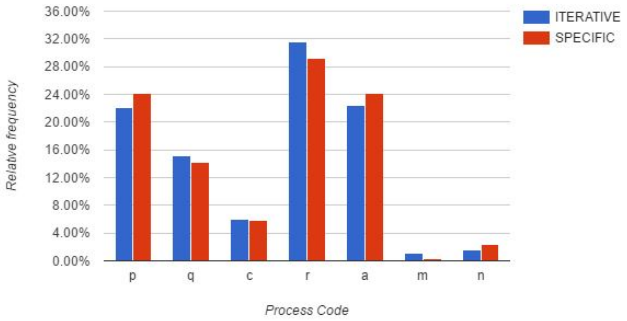


Fig. 2. Process codes frequencies

about the model ("m"). More in details, Figure 2 displays a clear unbalance between the number of confirmatory ("a" – 852 cases, 23.41%) and critical interactions ("c" – 214 cases, 5.88%). The low frequency of interactions classified as "environment" in the topic dimension (94 cases, 2.58% of the total) suggests that the study settings had a marginal impact on the approach followed by the subjects.

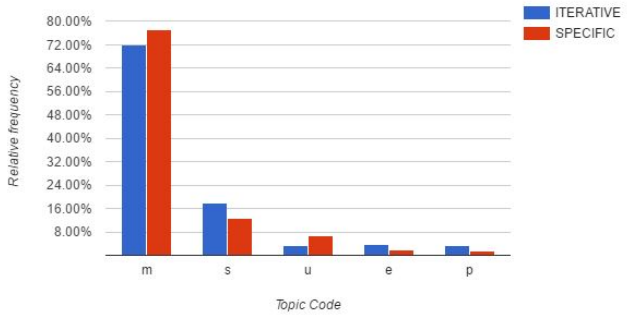


Fig. 3. Topic codes frequencies

We agree that aggregated data is hardly likely to provide additional information on the approach followed

by the participants. However, we can focus on specific subsamples in our data to support our answers to our RQs. More specifically, we select those interactions that have been classified as ”lack of knowledge” to identify the topics discussed in those specific statements. This provided us with information to support our answer to RQ2. Also, the comparison of the frequencies of both process and topic codes between the specific and iterative approach can help to explain how different requirements specifications affect the procedure followed by the subjects and the topics they discuss. We follow this approach to support our answer to RQ3.

We used the data from the coded transcriptions to get a clearer understanding of the extent to which different difficulties were present in the sessions we observed. We narrowed our focus to the 75 interactions that were classified as ”lack of knowledge” in the process coding. According to the definitions we provided in Table I, this category refers to those cases where the subjects communicated the inability to proceed in their solution because of the lack of a necessary piece of knowledge. Figure 4 shows the frequencies of the topic codes we observed in the subsample.

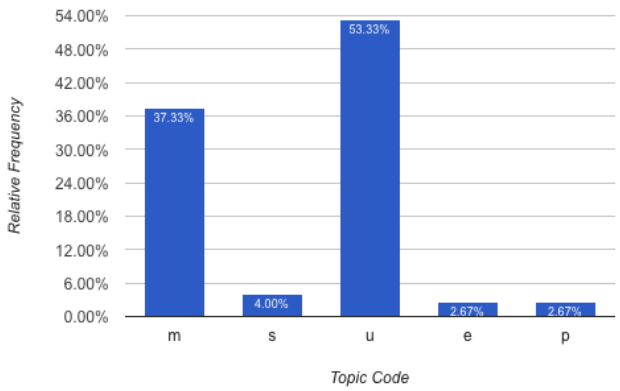


Fig. 4. Topic codes over Lack of Knowledge

The data displayed in Figure 4 is in line with the description we provided earlier in this paragraph, showing how UML syntax (40 cases) was by large the most common topic discussed in interactions where the subject admitted insufficient knowledge. Interactions related the model (28 cases) were in most cases expressing the inability to completely understand the runtime behavior of the elements in the model.

Finally, to better understand the general intentions of the students when discussing syntax rules, we looked at the frequencies of the process codes over the subsample containing the interactions related to UML syntax.

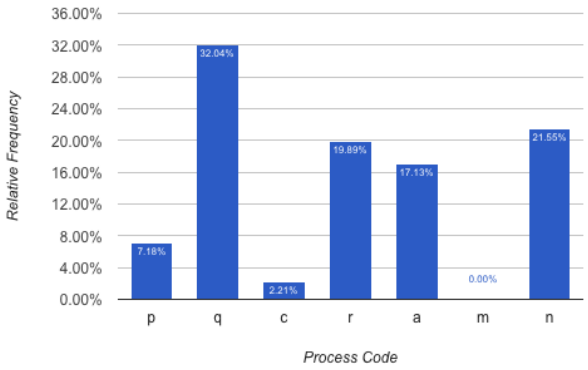


Fig. 5. Process codes in interactions related to UML syntax

Data in Figure 5 shows that when discussing syntax rules the students were often asking questions (32.04%) or expressing lack of knowledge (21.55%). Conversely, the figure shows how constructive suggestions (”proposing”) over this topic were limited to 7.18% of the total. Interactions of type ”mediate” were absent in this subsample as questions related to modelling or syntax were not allowed by the rules of the study.

To further investigate the differences we observed between the two scenarios, we compared the distributions of the interactions under the two scenarios we recreated in the study. We first divided the sample in two subsamples, one related to the iterative approach (from now on called *iterative sample*) and one related the specific approach (from now on called *specific sample*). The size of the iterative and specific samples are 1534 and 2105 respectively.

We used the Pearson’s chi-squared test to determine whether the differences between the distributions in the two subsamples could be considered statistically significant. The Pearson’s chi-squared test is a nonparametric test for analysis of categorical data, i.e. non-ordinal, that can be used for unpaired datasets from large samples, which made it ideal for our purposes [32]. Briefly, the Pearson’s chi-squared test (also called ”goodness-of-fit” test) is based on the assumption that the observed categorical variable follows a known distribution (reference distribution). This assumption allows the tester to use the relative frequencies–i.e. frequencies expressed as percentage of the total number of cases–in the reference distribution to compute the frequencies that are expected in samples that are extracted from the same population. The test then uses

Process Code	P_i	E_i	O_i	$O_i - E_i$	$(O_i - E_i)^2 \div E_i$
p	24.13%	370.20	338	–32.20	2.80
q	14.16%	217.16	233	15.84	1.15
c	5.80%	88.91	92	3.09	0.11
r	29.17%	447.45	484	36.55	2.99
a	24.09%	369.47	345	–24.47	1.62
m	0.29%	4.37	17	12.63	36.47
n	2.38%	36.44	25	–11.44	3.59
χ^2					**48.73**

TABLE VII
PEARSON CHI-SQUARED TEST STATISTICS FOR PROCESS DIMENSION

the differences between the observed and expected frequencies to compute the value of the test χ^2 statistic and determine whether those differences are statistically significant. The Pearson's chi squared test is often used to compare the distributions of a categorical variable over two samples. In these cases, one of the sample distributions act as the reference distribution. We performed two tests, each one dedicated to a coding dimension. In both tests we selected the distribution of the codes over the specific sample as the reference distribution.

Test 1: Specific vs Iterative approach – Process dimension

The intent of the first test was to determine whether the differences in the requirements specification documents would imply significant differences in the process followed by the subjects. In this case our observed variable is the process code we assign to the interactions, while the categories are all the codes we listed in Table I with the exception of "n/a". Below we define our null and alternative hypotheses:

$$H_0 : O_i = E_i$$
$$H_1 : O_i \neq E_i$$

where O_i is the frequency of the i-th category observed in the iterative sample, while E_i is the expected frequency of the same category. In common language, H_0 states that the number of observed frequencies in the iterative sample matches the expected values that were calculated based on the relative frequencies observed in the specific sample. Conversely, H_1 states that the frequencies differ significantly, implying a difference in the populations underlying the two samples.

The process codes includes 7 different categories, which translates in $7 - 1 = 6$ degrees of freedom (df). We set our alpha to 0.01 and obtain a critical value of the χ^2 statistic equal to 16.81. This value is then compared with the observed test statistic to determine whether the differences in the distributions can be considered to be statistically significant.

Table VII shows the values we obtained in our calculation of the test χ^2 statistic. With P_i we indicate the relative frequencies observed in the specific sample, while E_i represents the frequencies of each category that we expect to observe in the iterative sample. The values of E_i were obtained by multiplying the corresponding relative frequency in the specific sample by the size of the iterative sample ($n_{it} = 1534$). O_i values indicate the actual frequencies observed in the iterative sample. The resulting value of the χ^2 statistic is 48.73. Since the test statistic is above the critical value of 16.81 we can reject the null hypothesis at a 0.01 level of significance and conclude that the distribution of the interactions over the process dimension changes with the type of requirements specifications provided to the subjects.

Values in Table VII, column 5 display the differences between the observed and expected frequencies in the iterative sample for each process code category. We see that in the iterative approach we observed less proposing, acknowledgement and lack of knowledge than in the specific approach. Conversely, the iterative sample displayed higher frequencies for categories like questioning, criticising, reasoning and mediating. Values in column 6 say that the discrepancy between the two samples is mainly due to the differences observed in the category "mediate". Because of the low frequency of "mediate" we observed in the specific sample (6 cases, 0.29%), small deviations in this category have large impact on the test statistics. Any interpretation of these results should therefore take this factor into account, especially in consideration of the nature of the data composing the sample, i.e. subjective classifications of verbal interactions.

Topic Code	P_i	E_i	O_i	$O_i - E_i$	$(O_i - E_i)^2 \div E_i$
m	77.29%	1185.66	1105	–80.66	5.49
s	12.54%	192.39	273	80.61	33.78
u	6.75%	103.48	50	–53.48	27.64
e	1.85%	28.42	55	26.58	24.86
p	1.57%	24.05	51	26.95	30.21
		χ^2			**121.97**

TABLE VIII
PEARSON CHI-SQUARED TEST STATISTICS FOR TOPIC DIMENSION

Test 2: Specific vs Iterative approach – Topic dimension

With our second test we intend to check whether different requirements specifications implied changes in the topics discussed by the subjects. We proceed following the same approach we used in our test over the process codes. Again, our null and alternative hypotheses are:

$$H_0 : O_i = E_i$$
$$H_1 : O_i \neq E_i$$

where O_i is the frequency of the i-th topic code category observed in the iterative sample, while E_i is the expected frequency of the same category. H_0 states that the number of observed frequencies in the iterative sample matches the expected values that were calculated based on the relative frequencies observed in the specific sample. Conversely, H_1 states that the frequencies differ significantly, meaning that the change of scenario affected the topics discussed by the subjects.

The topic codes include 5 categories, which means the degrees of freedom are equal to $5 - 1 = 4$. We select an alpha of 0.01 and obtain a critical value of the χ^2 statistic equal to 13.28. Table VIII shows our calculation of the test χ^2 statistic.

The resulting value of the χ^2 statistic is 121.97. Again, the test statistic is clearly above the critical value of 13.28, meaning that we can reject the null hypothesis at a 0.01 level of significance and conclude that the distributions of the interactions over the topic dimension over the two sub-samples are significantly different.

The values displayed in Table VIII show that in the iterative approach subjects tended to focus their discussion more on system requirements, procedure and study environment, while interactions concerning the model or UML syntax were less frequent than expected. Values in column 6 show how all categories displayed large deviations from their expected relative frequencies, with categories ”s” (system) and ”p” (procedure) ranking first and second respectively.

V. DISCUSSION

In Section IV we presented the data we collected during this study. We used the frequencies of the interactions over the process and topic dimensions to investigate the approach of the students. In this section we describe the common themes we identified over different sessions. These descriptions are used together with the answers from the post-study interviews to provide an explanation of our findings.

Modelling Approach

During the study sessions we closely observed the activities of the students in a first attempt to identify common themes among different groups. All groups displayed a high level of participation and involvement. Subjects used the time at their disposal to actively cooperate to create the required models. We observed different groups follow various types of approaches. For example, in one particular case (G2) both subjects started by creating two separate solutions and then produced a final model by combining them. In some cases we observed only one subject drawing on the board and the other reading the requirement specifications (G1), while other groups opted for more flexible roles.

The groups also followed a different sequence of steps when it came to introducing UML elements in the state machines they modeled. One group (G4) started by identifying all states, and then proceeded with the definition of the necessary logic to handle the transitions between them. Two groups (G2 and G5) seemed to follow a more incremental approach, starting from the entry state and then proceeding transition by transition. In the remaining two cases (G1 and G3) we observed the subjects abandon their solutions in more than one instance, which in some cases made it hard for us to interpret their plans. One thing that we found particularly curious in the post-study interviews was that students could not describe

their own process or strategy correctly. When asked to describe their approach to modelling during the sessions, all groups agreed that they would first identify the states and then define the transitions. As we mentioned, this description contradicts the actual steps we observed during the sessions.

Due to the discrepancies we just described, the cases we observed in this study did not present enough similarities to allow us to identify clear patterns in terms of adopted strategy (RQ1).

Common difficulties

In order to present the common difficulties encountered by the groups we first need to discuss their solutions. In this discussion we distinguish between the functionality of the model and the correctness of the UML syntax. More specifically, in Section III we described how the students were allowed to declare their own syntax whenever they would feel stuck in the design of their solution. Therefore, when we refer to functionality we do not primarily consider the correctness of the UML syntax but we focus on the behaviour of the model as intended by its designers.

We rated three out of ten models as completely functional. Two of these models were delivered by the same group (G4), which was also the fastest group in completing their solutions (see Table V). The other functional solution was designed by G5 from the URS document on the elevator system. Three of the remaining models presented the same issue preventing their functionality, which was related to a wrong use in the ”exit node” included in their state machines. We judged the four remaining models as non functional as they did not include all the necessary elements to ensure the desired behaviour in the model. The most common mistakes in these solutions can be reconducted to either missing transitions or failing in updating variable values when transitioning between states.

The models also presented some issues related to the use of UML elements. For example, two groups made extensive use of boolean variables and guards, which were used to handle cases where triggers and events were required. More specifically, in these cases the subjects defined a boolean variable for each functionality provided by the system. When the user requests a functionality to start, the corresponding boolean variable is set to true and all the others were set to false. This specific anti-pattern generated the need of continuously reset the variables values in the model, therefore making the extensibility of the model heavier and more complex. Other common mistakes were related to the use of choice nodes, nested states and state behaviour. In this sense, our results correlate with the findings of Stikkolorum et al. [14], who found that student subjects experienced struggles in using the correct UML class diagram elements for a particular solution.

In addition to the flaws we found in the final solutions, we identified two topics that drew considerable attention during the modelling sessions. The first one concerned the modelling of events that were required to affect the operativity of the system only when it is in a specific state. More in details, the SRS we created for the elevator and the drying machine included a complete description of the cases in which user input should be handled or ignored by the system. An example from the elevator SRS can provide a better understanding:

”3. If a passenger requests the emergency stop and the elevator is moving, the elevator shall stop immediately.”
”4. If a passenger requests the emergency stop and the elevator is not moving, nothing happens.”

We noticed that requirements specifying “nothing happens” often raised questions and generally captured the attention of the students. The following quote was extracted from our transcriptions:

”So how do you model that? If the passenger requests the emergency stop and the elevator is not moving, nothing happens.”

In particular, the students were often discussing whether it would be necessary to display ignored events in the model by drawing reflexive transitions[11] in those states that were not affected. If on the one hand, the inclusion of such transitions does not always compromise the functionality of the model, on the other it unnecessarily increases its complexity. During this study we observed all groups, at least briefly, engage in this type of discussion. However, this flaw was found in only one of the five solutions provided to SRS specifications.

Another recurring theme in the discussions was related to the scope of the model. For example, the requirements specifications for the elevator system described responses to events coming from different external actors, namely users and an external control system. Users were further classified as passengers (i.e. users inside the elevator) or callers (i.e. users outside

[11]Transition from a state to itself

the elevators). In three different cases (G1, G2 and G3) we observed the subjects discuss whether external actors should be included in the scope of the model. Again, we provide one example from our transcriptions:

"I mean, do we ... do we ... in a state machine do we model different entities as like distinct objects?"

Regardless of the number and identity of the external actors interacting with the system, a viable solution in this type of requirements would consist in limiting the scope of the model to the system (the elevator) and in treating the interactions of the external actors as events. We observed G1, G3 and G5 discuss this topic extensively during the modelling sessions.

The difficulties related to UML syntax were also a common theme in the post-study interviews. Six out of ten participants indicated the selection of the correct state machine element as their main difficulty. According to the students, this often put them in the position of not being able to express their thoughts. As one participant described it:

"I felt like I was making a lot of mistakes. I was not able to sometimes express what I was thinking."

The frequencies displayed in Figure 4 and 5 provide ulterior support to our findings. Figure 4 showed that interactions classified as "lack of knowledge" were the most common in interactions concerning UML syntax (53.33%). This number should also be interpreted by taking into account the conservative definition of the code "u" we provided in Table II. More specifically, interactions containing any reference to the model (or any element within it) were coded as "m", while the use of the code "u" was limited to those cases where syntax rules were described in a more abstract context. As a consequence, many of the cases that are classified as "model" (37.77%) can still be interpreted as related to syntax issues. As a fact, most of these interactions reflected the inability of the subjects to completely understand the runtime behavior of the model they created. Again, the most common elements discussed in these interactions were related to reflexive transitions, choice nodes, nested states and state behaviour.

The description we provided above reports difficulties related to conventions (e.g. ignoring events, handling interactions with external actors) and syntax rules of UML state machines. Our interpretation is that these difficulties were often related to specific cases, and rarely reflecting a poor understanding of the overall behaviour of the model or to state machines in general. In addition, we observed that the subjects were often able to identify and discuss the issues they were encountering. This suggests that they would probably be able to overcome these issues in a context where they had access to the necessary piece of information (e.g. the cheat sheet). Table IV shows that all participants had academic experience in software modelling. At the same time, the rightmost column shows that subjects declared not to be frequently involved in activities such as creating and reading models, with seven subjects out of ten claiming they would engage in such activities on a yearly basis. We believe that this can partially explain the uncertainties we observed during the sessions.

Specific vs. Iterative approach

During this study we reproduced two distinct scenarios with the intention to explore how different requirements specification formats can affect the modelling process of the students. On the one side we created vague URS requirements and allowed for more frequent interactions with the researchers (iterative approach), and on the other we provided detailed SRS and limited the chances for the students to request for additional details (specific approach).

In Section IV we mentioned that none of the groups took advantage of the second break in the iterative approach, with Table V showing that all solutions were completed within the first two modelling iterations. Table IX shows the number of questions related to system requirements that each group asked to the researchers during the break between the two iterations. The cell marked with the dash symbol denotes a break that did not take place as G4 made only use of one iteration.

	Approach	
Group	**Specific**	**Iterative**
G1	1	4
G2	3	6
G3	1	2
G4	0	-
G5	1	1
Total	**6**	**13**

TABLE IX
QUESTIONS ASKED BETWEEN ITERATIONS

If on the one side Table IX displays a higher number of total questions asked during the iterative approach, on the other the values from G4 and G5 do not allow us to

make any sort of generalization. Also, during the iterative sessions we observed all groups to make use of assumptions as a response to the vagueness of URS. Even if this reaction was not completely unexpected, what we found interesting was that in many cases students did not decide to verify the correctness of these assumptions with the researchers. This aspect was more pronounced in some groups than others. G4 was definitely the clearest case, with the subjects completing their model before the end of the first iteration without recurring to any interaction with the researchers. G5 followed a similar procedure, asking only one question under both scenarios.

We report that the formulation of assumptions did not always lead the subjects to a positive outcome. More specifically, in different cases we observed the subjects discuss aspects of the system that were outside the scope of the requirements. Two clear examples are G3 and G5, which spent considerable effort discussing the design of the mechanism for opening and closing the doors of the elevator system, while no door or any related functionality was mentioned in the requirements specifications. In these two cases the introduction of such mechanism in the model did not compromise the functionality of the solutions. A different example is provided by G1, which initially assumed that the dryer system could abort a preset 'program' while it was running. We clarify that this specific assumption was incorrect as it was contradicting the URS for the dryer system. However, G1 addressed the researchers to verify this specific assumption during the first break and eventually modified their model according to the new information they collected.

In Section IV we used the Pearson's chi-squared test to check whether the different settings in the two scenarios affected the way subjects interacted with each other. The use of the statistical tests on the frequencies of the interactions is to be intended as complementary to the qualitative description we provide in this paper. The intent here is not to provide measurements or forecasts for interactions outside our sample, but rather to check whether the coded transcriptions would support our understanding of the differences we observed between the two scenarios. Our results reported significant differences in the distributions of both process and topic codes, suggesting that the use of different requirements specification formats produced a change in both the procedure and the topics discussed by the subjects. More specifically, in Section IV we showed that the vague URS affected the modelling approach of the students by fostering more mediate, reasoning and questions about the system. On the one side the higher frequency of mediate reflects more frequent interactions with the researchers, and on the other the higher frequencies of reasoning and questioning about the system quantify the additional effort spent by the subjects in formulating assumptions on the system requirements.

The comparison of our observations with the answers provided to the post-study interviews provides relevant insight. More specifically, subjects seemed to generally recognize the increased use of "mediate" under the iterative approach. When asked if the textual requirements specifications were clear, all students answered positively. However, when we asked at which point in time they felt like they had a complete understanding of the system, seven out of ten students specified that SRS were clear from the very beginning, while URS needed to be integrated with further details elicited from the customer between modelling iterations. The general idea was that the URS became completely clear after the first break. Only subjects from G4 seemed to recognize the use of assumptions that they made under the iterative approach. As P7 stated:

"The thing is, when you get it in a more abstract form, it's more up to yourself to fill in the blanks."

The fact that this aspect was mentioned only by one student suggests that subjects did not always recognize the relevance of the assumptions that were made and of their consequent design decisions. This can explain why we rarely observed the subjects verify the correctness of such assumptions during the breaks.

VI. Validity Threats

Maxwell [33] discusses the main threats to validity for qualitative research. According to this perspective, we identify the following main threats to this study.

Descriptive validity concerns the veracity and the completeness of the descriptions provided by the researchers. Although we agree with Maxwell that "no account can include everything" [33], in this paper we tried to provide a complete discussion of the aspects that we considered most relevant to our RQs. In addition, the audio and video recordings from the study sessions allow access to the necessary information to resolve any potential disagreement on this specific aspect.

The use of the coded transcriptions constitutes a threat to the interpretive validity [33] of this study. The concept of interpretive validity is related to reliability. According to Maxwell, interpretive validity is relevant to those

studies that aim to understand a phenomenon from the perspective of the participants, and more specifically when the researcher provides an interpretation of the intentions behind the actions and statements of the subjects [33]. In this sense, the validity of this study depends on the extent to which different researchers provide different interpretations of the recorded interactions. To mitigate this risk we used the Krippendorff's alpha to measure the inter-rater agreement and proceeded iteratively until reliability of the data was achieved.

Regarding generalizability [33], i.e. external validity, the limited number of sessions we conducted do not allow for a wide generalization of our findings. However, regardless of the reduced scope and because of the exploratory nature of the study, this paper provides a contribution of "rich insight" [34]. In other words, we intend to contribute to the existing body of knowledge on requirements modelling [15], and therefore, the conclusions we draw should be considered as a starting point for future research.

VII. Conclusions

In this paper we described a case study directed to investigate the approach of students to modelling of behavioural requirements. We followed an exploratory approach, where data analysis was carried out in parallel with data collection procedures. This allowed us to build on our findings in an iterative manner. This process resulted in a refinement and extension of the set of codes originally inherited from the replicated study [12].

Although we could not identify a common strategy over the five groups we observed, we were able to spot patterns that resulted useful in explaining our answers to RQ2 and RQ3. We observed that the most common difficulties were related to misuse or missing knowledge of specific elements of UML state machines, such as choice nodes, nested states and state behaviour. Also, we described how handling of external signals and modelling of external actors represented two recurring topics in the discussions and reasoning process of the subjects.

Finally, we used both URS and SRS to investigate whether different types of requirements specifications affect the procedure of the subjects. Although we observed the students seeking a higher customer involvement when using vague specifications, we also report a common use of assumptions in reaction to the informational gap recreated in the URS. Our qualitative description is supported by the statistical analysis of the coded interactions, which shows how students spent more effort in discussing system requirements under the iterative approach. The analysis of the post-study interviews showed that students rarely recognized the relevance of the assumptions they used to overcome the vagueness in the requirements specifications. As a consequence, assumptions were rarely documented, criticized or verified with the researchers.

In summary, the differences we observed under the two approaches suggest that the use of vague requirements can lead to different outcomes in terms of requirements modelling. More specifically, vague requirements can be beneficial as they stimulate the modeller to formulate assumptions on relevant requirements that may have not been considered by the customer. However, our insight also highlights some related risks, especially in those cases where assumptions are not questioned. This suggests a need to somehow document, identify and question the assumptions that are formulated when modelling requirements specifications.

Acknowledgement

We would like to thank our academic supervisor, Grischa Liebel, for his help and guidance during this thesis work. We would also like to thank all the students who participated in our study.

References

[1] F. Brooks, "No Silver Bullet: Essence and Accidents of Software Engineering," *Computer*, vol. 20, no. 4, pp. 10–19, Apr. 1987.

[2] B. W. Boehm *et al.*, *Software engineering economics*. Prentice-hall Englewood Cliffs (NJ), 1981, vol. 197.

[3] J. Helming, M. Koegel, F. Schneider, M. Haeger, C. Kaminski, B. Bruegge, and B. Berenbach, "Towards a unified requirements modeling language," in *Requirements Engineering Visualization (REV), 2010 Fifth International Workshop on*. IEEE, 2010, pp. 53–57.

[4] I. J. Jureta, A. Borgida, N. A. Ernst, and J. Mylopoulos, "Techne: Towards a new generation of requirements modeling languages with goals, preferences, and inconsistency handling," in *Requirements Engineering Conference (RE), 2010 18th IEEE International*. IEEE, 2010, pp. 115–124.

[5] G. Liebel and M. Tichy, "Comparing Comprehensibility of Modelling Languages for Specifying Behavioural Requirements." in *HuFaMo@ MoDELS*, 2015, pp. 17–24.

[6] S. Abrahao, C. Gravino, E. Insfran, G. Scanniello, and G. Tortora, "Assessing the effectiveness of sequence diagrams in the comprehension of functional requirements: Results from a family of five experiments," *IEEE Transactions on Software Engineering*, vol. 39, no. 3, pp. 327–342, 2013.

[7] M. C. Otero and J. J. Dolado, "Evaluation of the comprehension of the dynamic modeling in UML," *Information and Software Technology*, vol. 46, no. 1, pp. 35–53, 2004.

[8] E. Duala-Ekoko and M. P. Robillard, "Asking and answering questions about unfamiliar APIs: An exploratory study," in *Proceedings of the 34th International Conference on Software Engineering*. IEEE Press, 2012, pp. 266–276.

[9] J. Sillito, G. C. Murphy, and K. De Volder, "Asking and answering questions during a programming change task," *IEEE Transactions on Software Engineering*, vol. 34, no. 4, pp. 434–451, 2008.

[10] N. Miyake, "Constructive interaction and the iterative process of understanding," *Cognitive science*, vol. 10, no. 2, pp. 151–177, 1986.

[11] T. Boren and J. Ramey, "Thinking aloud: Reconciling theory and practice," *IEEE transactions on professional communication*, vol. 43, no. 3, pp. 261–278, 2000.

[12] G. Liebel, "Exploratory Quasi-Experiment: Requirements Modelling," 2017.

[13] F. J. Shull, J. C. Carver, S. Vegas, and N. Juristo, "The role of replications in empirical software engineering," *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[14] D. R. Stikkolorum, T. Ho-Quang, B. Karasneh, and M. R. Chaudron, "Uncovering Students' Common Difficulties and Strategies During a Class Diagram Design Process: an Online Experiment." in *EduSymp@ MoDELS*, 2015, pp. 29–42.

[15] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to advanced empirical software engineering*. Springer, 2008, pp. 285–311.

[16] J. Sillito, K. De Voider, B. Fisher, and G. Murphy, "Managing software change tasks: An exploratory study," in *Empirical Software Engineering, 2005. 2005 International Symposium on*. IEEE, 2005, pp. 10–pp.

[17] D. E. Damian, A. Eberlein, M. L. Shaw, and B. R. Gaines, "An exploratory study of facilitation in distributed requirements engineering," *Requirements Engineering*, vol. 8, no. 1, pp. 23–41, 2003.

[18] M. N. Marshall, "Sampling for qualitative research," *Family practice*, vol. 13, no. 6, pp. 522–526, 1996.

[19] OMG, "Omg Unified Modeling Language Specification, version 1.3, june 1999," http://www.omg.org, 1999.

[20] M. C. Otero and J. J. Dolado, "An initial experimental assessment of the dynamic modelling in UML," *Empirical Software Engineering*, vol. 7, no. 1, pp. 27–47, 2002.

[21] S. Kuske, "A formal semantics of UML state machines based on structured graph transformation," in *International Conference on the Unified Modeling Language*. Springer, 2001, pp. 241–256.

[22] D. Budgen, A. J. Burn, O. P. Brereton, B. A. Kitchenham, and R. Pretorius, "Empirical evidence about the UML: a systematic literature review," *Software: Practice and Experience*, vol. 41, no. 4, pp. 363–392, 2011.

[23] I. Sommerville, *Software Engineering: (Update) (8th Edition) (International Computer Science)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2006.

[24] J. R. Fraenkel, N. E. Wallen, and H. H. Hyun, *How to design and evaluate research in education*. McGraw-Hill New York, 1993, vol. 7.

[25] J. Linaker, S. M. Sulaman, M. Höst, and R. M. de Mello, "Guidelines for Conducting Surveys in Software Engineering v. 1.1," 2015.

[26] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on software engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[27] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical software engineering*, vol. 14, no. 2, p. 131, 2009.

[28] D. R. Thomas, "A general inductive approach for analyzing qualitative evaluation data," *American journal of evaluation*, vol. 27, no. 2, pp. 237–246, 2006.

[29] A. Soller, "Supporting social interaction in an intelligent collaborative learning system," *International Journal of Artificial Intelligence in Education (IJAIED)*, vol. 12, pp. 40–62, 2001.

[30] M. J. Baker, "Argumentation and constructive interaction," *Foundations of argumentative text processing*, vol. 5, pp. 179–202, 1999.

[31] K. Krippendorff, "Reliability in content analysis," *Human communication research*, vol. 30, no. 3, pp. 411–433, 2004.

[32] A. Agresti and M. Kateri, *Categorical data analysis*. Springer, 2011.

[33] J. Maxwell, "Understanding and validity in qualitative research," *Harvard educational review*, vol. 62, no. 3, pp. 279–301, 1992.

[34] G. Walsham, "Interpretive case studies in IS research: nature and method," *European Journal of information systems*, vol. 4, no. 2, p. 74, 1995.

APPENDIX A

Elevator System - Iterative Approach

We (as the customer) would like to have an elevator system for our office building. Users inside the elevator should be able to select the floor they wish to go to. Also, they should be able to stop the elevator in case of emergency. Users outside the elevator can call the elevator to their current floor on the press of a button. All requests different from the emergency-stop should be ignored while the elevator is moving. Finally, maintenance personnel should be able to restart the elevator from the external control system when the elevator is in emergency-stop.

Elevator System - Specific Approach

We use the terms *passenger* and *caller* to distinguish between users inside and outside the elevator respectively.

1. If a passenger requests the elevator to move to a floor and the elevator is not moving, the elevator shall move to the selected floor. If the passenger selects the same floor she is on, nothing happens.
2. If a passenger requests the elevator to move to a floor and the elevator is moving, nothing happens.
3. If a passenger requests the emergency stop and the elevator is moving, the elevator shall stop immediately.
4. If a passenger requests the emergency stop and the elevator is not moving, nothing happens.
5. If the elevator is in ‘emergency-stop’ and a reset request arrives from the central elevator control system, the elevator will move to the next lower floor.
6. If a caller requests the elevator to move to her current floor and the elevator is not moving, the elevator shall move to the caller’s floor.
7. If a caller requests the elevator to move to her current floor and the elevator is moving, nothing happens.
8. If a caller requests the elevator to move to her current floor and the elevator is in ‘emergency-stop’, nothing happens.

Tumble Dryer System - Iterative Approach

We (as the customer) would like to have a tumble dryer system that provides two preset programs: ‘humidity’ and ‘timer’. The humidity program runs until the values of the readings from a humidity sensor fall below a minimum threshold. The timer program starts with a default duration and stops when the timer expires. Users should be able to increase and decrease the value of the timer while the timer program is running.

Users can pause a program that is currently running. Finally, users should also be able to terminate or resume a paused program.

Tumble Dryer System - Specific Approach

1. If the user requests to start the humidity program while the dryer is ready[1], the system starts operating in humidity mode.
2. If the user requests the dryer to start the humidity program while the dryer is not ready, nothing happens.
3. If the humidity program is running and the sensor detects a level of humidity below $1g/m^3$, the program is terminated.
4. If the user requests to start the timer program while the dryer is ready, the system starts operating in timer mode with a duration of 30 minutes.
5. If the user requests the dryer to start the timer program while the dryer is not ready, nothing happens.
6. If the timer program is running and the timer expires, the program shall be terminated.
7. If the user requests to increase the timer and the timer program is running, the timer shall be increased by 1 second.
8. If the user requests to decrease the timer and the timer program is running, the timer shall be decreased by 1 second.
9. If the user presses the pause button while a program is running, the program shall be paused.
10. If the user presses the pause button while the dryer is ready, nothing happens.
11. If the user presses the resume button while a program is paused, the program shall be resumed.
12. If the user presses the terminate button while a program is paused, the program shall be terminated.

[1] No program is currently running or paused.

APPENDIX B

Name/ID:

Exploratory Requirements Modelling

Pre-study Questionnaire

1. What is your gender? _____

2. What is your highest degree?

a. High school diploma
b. Bachelor diploma
c. Master diploma
d. PhD diploma
e. Other (please specify):_____

3. Please list relevant courses that you took in Requirements Engineering, Modelling or UML diagrams. Please provide the institution and the name of the course!

a. _____
b. _____
c. _____
d. _____

4. How many years of **industrial** experience do you have in ...

a. … developing software/coding ___
b. … software modelling ___
c. … requirements engineering ___

5. How would you judge your knowledge on the following topics?

	No Knowledge	Basic [1] Knowledge	Intermediate [2] Knowledge	Expert [3] Knowledge
UML	◯	◯	◯	◯
State Machines	◯	◯	◯	◯

[1] I can understand and produce simple models with a few elements.
[2] I can understand and produce models using advanced language concepts.
[3] I can understand and produce highly complex models.

Name/ID:

6. How would you judge your knowledge on the following domains?

	No Knowledge	Basic [4] Knowledge	Intermediate [5] Knowledge	Expert [6] Knowledge
Embedded Systems	○	○	○	○
Vertical Transportation Systems	○	○	○	○
Requirements Engineering	○	○	○	○

7. How often do you do the following tasks?

	Less than yearly	Yearly	Monthly	Weekly or more
Programming	○	○	○	○
Create Models	○	○	○	○
Read Models	○	○	○	○

[4] I know the basic terminology, but do not have any practical experience.
[5] I have advanced theoretical knowledge and/or limited practical experience.
[6] I have extensive practical and theoretical knowledge of the subject.

APPENDIX C

Name/ID:

Exploratory Requirements Modelling

Post-study Questionnaire

	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
I had enough time to model the system.	○	○	○	○	○
In the end, the requirements were clear.	○	○	○	○	○
In the beginning, the requirements were clear.	○	○	○	○	○
It was easy to model the requirements (overall).	○	○	○	○	○
I am confident in the functionality of the model.	○	○	○	○	○

1. What was easy in the modelling process?

2. What was difficult in the modelling process?

3. What kind of information were you lacking?

4. How did you approach the modelling process (how would you describe what you did)?

5. Were your difficulties mainly related to a lack of domain knowledge or a lack of modelling knowledge?

6. What would you change to improve the quality of the models you produced?

7. From what point in time would it have been useful to use a CASE tool? For what purpose?

8. What are the positive and negative effects of pair modelling?

9. At what point in time did you fully understand the stakeholders need (requirements)?

10. Is there anything you would like to add?

Exploratory Requirements Modelling

UML Finite State Machines Notation Cheat Sheet

UML Finite State Machines (FSM) diagrams are used to model object behavior. FSM display the lifecycle of an object: what events it experiences, its transitions and the states between these events[1].

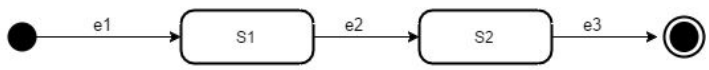
States, Events and Transitions

A **state** describes the behavior of the system at a point in time - i.e. between the occurrence of events. FSM diagrams can specify an **entry** (initial) and **exit** (final) state.

A **transition** between two states A and B defines the conditions that must hold for the object to move from state A to state B.

An **event** is a noteworthy occurrence, i.e. something that happens that may trigger a transition. An event can be internal (e.g. system message), external (e.g. user action) or a time event.

Figure 1 - States, Events and Transitions



●	entry (initial) state
◉	exit (final) state
S1, S2	states
e1, e2, e3	events

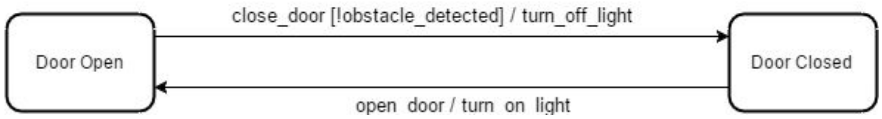
[1] Larman C, Applying UML and patterns. Upper Saddle River, N.J. Prentice Hall PTR, 2005.

Guards and Effects

A **guard** is a condition that must evaluate to true in order for the transition to happen. An **effect** describes a behavior that is fired by a transition.

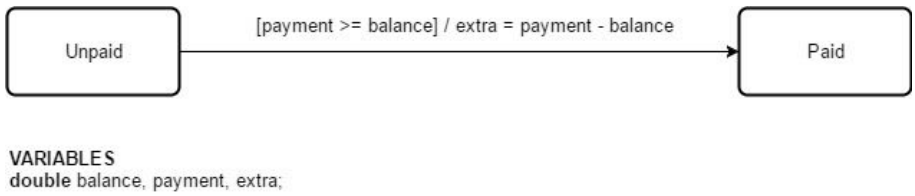
The notation for a transition including guards and effects is **event [guard] / effect.**

Figure 2 - Guards and Effects - Sliding doors system



FSM transitions can be defined in terms of variables. Variables can be used to define guards, while effects can be used to update variable values.

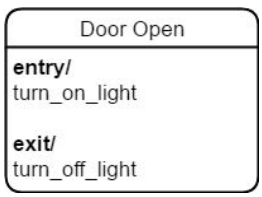
Figure 3 - Variables in Guards and Effects



State Behavior

States can specify the behavior that is executed when the state is entered (**entry/**), while it is active (**do/**) or when it is exited (**exit/**).

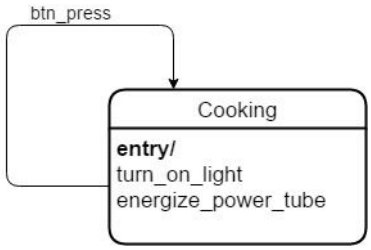
Figure 4 - State Behavior



Reflexive Transitions

Reflexive transitions connect a state to itself. When a reflective transition is taken, the state behavior is executed.

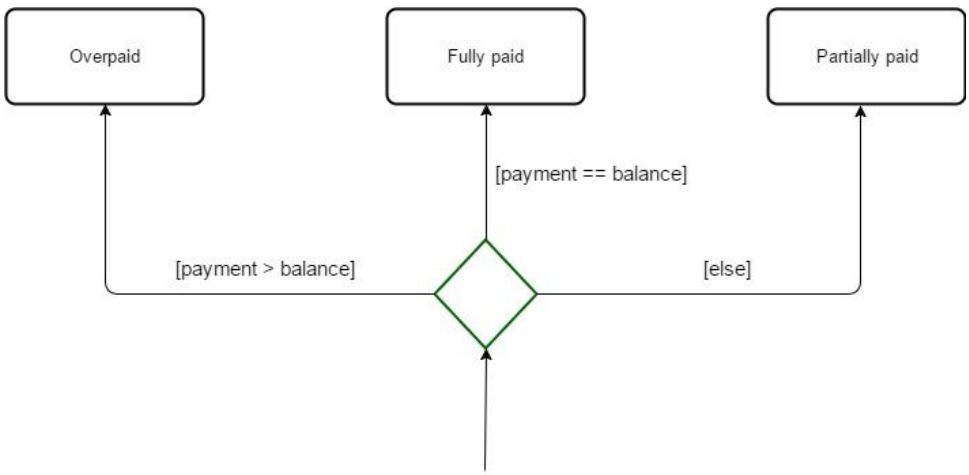
Figure 5 - Reflective Transitions



Choice nodes

Choice nodes can be used to model cases where transitions are based on multiple conditions.

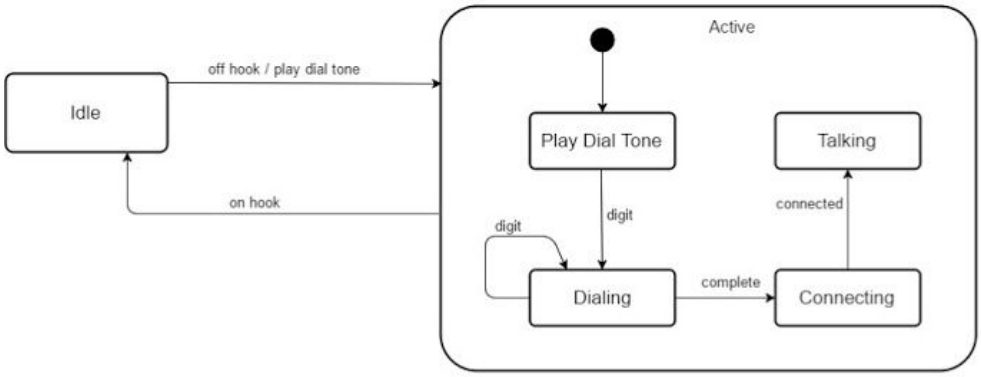
Figure 6 - Choice nodes



Nested States

States are allowed to contain substates. Substates inherit the transitions of the parent state - i.e. the container[2].

Figure 7 - Nested States



[2] Larman C, Applying UML and patterns. Upper Saddle River, N.J. Prentice Hall PTR, 2005.

APPENDIX E

Minute	Subject	Statement	Comment	Process	Topic
	s2	But are you...? The timer program starts with a default duration		r	s
	s1	Yeah, but we will have duration.		p	m
	s1	Int or something? Minutes?		q	m
	s2	Yeah, duration yeah.		a	m
	s2	Seconds		p	m
	s1	mmmm yeah...		a	m
	s2	If you have seconds you could also use it to tick down		p	m
	s1	Yeah, sure, seconds		a	m
	s1	programming... assuming that programming languages are more suitable for...	writes down variable declaration on the whiteboard	r	m
03:00	s1	yeah sure, so we do not need to care about the default. We can assume it will be coded somewhere		p	m
	s2	But here mmmm... obviously you can return at some point?	mimics transition on the board	q	m
	s1	yeah		a	m
	s2	but now we need some...	draws	n/a	n/a
	s2	So... what can we write here?		q	m
	s2	We can write a trigger, "do_humidity_program"		p	m
	s1	Yeah		a	m
	s2	start_humidity_program		p	m
	s1	Should there be a guard?		q	m
	s1	If it is humidity or not?		q	m
	s1	Doesn't say, it does not say it. That is why this is abstract.	Points at the requirements specifications.	r	s
	s2	If the tumbledryer... it always starts.		r	s
	s2	It takes several minutes, even if it is dried.		r	s
	s1	It's maybe our design decision		r	m
	s2	I'll just say start...	declares trigger on the whiteboard	p	m
04:00	s1	for now that's just start.		a	m
	s1	And this is an event or a guard.		p	m
	s1	It says... "sensor reading is under"	reading requirements document	r	s
	s2	Exactly so...		a	s
	s2	So sensor value...	drawing	p	m
	s1	...below minimum threshold		p	m
	s1	and some method call maybe		p	m
	s2	Finish, finish program... finish	drawing	p	m
	s2	There must be some stop sequence		p	m
	s2	And here, start...		p	m
	s1	and when duration sec		p	m
	s2	is equal to...		p	m
	s2	Duration is?	drawing	q	s
05:00	s1	Zero		p	s
	s2	Smaller-equal zero	writing guard on transition	p	m
	s1	then stop... timer... Finish	drawing	p	m
	s2	timer program... Awesome!	drawing	p	m
	s1	But here, did you do it?		q	m
	s1	Mmmmm... increase and decrease.		p	m
	s1	But there is nothing there		r	m
	s2	No, there is nothing there. It's just here.		r	m
	s2	Here we could just change the duration.		p	m